Napór czytelników-właścicieli ZX SPECTRUM na mikroKLAN nie tylko nie słabnie, ale wręcz nasila się. Dlatego też poświęcamy tyle miejsca temu mikrokomputerowi w tym numerze. O ile wielbicieli sprzętu chyba w pełni usatysfakcjonuje publikowany schemat ZX SPECTRUM, to dla zadowolenia tych co „grzebią się" w oprogramowaniu przygotować musieliśmy coś równie atrakcyjnego. Nie możemy wprawdzie opublikować „zdisasemblowanego" ROM-u (zajmuje to wcale nie małą książkę), ale za to publikujemy pierwszą część materiału wyjaśniającego jak uzupełniać to co jest we wspomnianym ROM-ie.

Możliwość rozszerzenia podstawoweko zestawu instrukcji może okazać się wyjątkowo użyteczna przy wszelkich niestandardowych zastosowaniach. Dotyczy to zarówno przystawek sprzętowych, jak i pewnych zastosowań programowych, w których BASIC nie wystarcza, lecz z różnych powodów (miejmy nadzieję, że nie tylko z lenistwa) nie chcemy sięgać po kompilatory innych języków.

Jak rozszerzyć BASIC ZX SPECTRUM

Wśród wad, które zarzuca się ZX SPECTRUM, dosyć często wymieniany jest ograniczony zestaw instrukcji. Oczywiście niedociągnięcia nie czekały zbyt długo na poprawienie. Powstały odpowiednie programy — na przykład — BETA BASIC z jego ostatnią wersją V 1.8. czy SUPERCODE 2. Naprzeciw oczekiwaniom klientów wyszła również sama firma Sinclair, umożliwiając przy wykorzystaniu INTERFACE 1 tworzenie nowych instrukcji, definiowanych przez samego użytkownika.

Okazuje się, że jest to możliwe również bez tej kosztownej przystawki — i to dla obu wersji komputera: 16 K i 48 K. Koncepcja rozszerzania zestawu instrukcji jest stosunkowo prosta, choć jej praktyczna realizacja niestety do najprostszych nie należy.

Tym, którzy po lekturze tego artykułu zapragnęliby dodać kilka funkcji użytkowych do swego komputerka, radziłbym wcześniej zapoznać się z możliwościami wspomnianego wcześniej programu BETA BASIC. Z pewnością łatwiej będzie zdobyć ten program, niż „na piechotę" wprowadzać duże ilości kodu maszynowego. Opisana metoda może natomiast przydać się dla uzupełnienia zestawu instrukcji w języku BASIC ZX SPECTRUM o nietypowe funkcje (np. graficzne). Być może niektórzy już domyślają się, jak można rozszerzyć zestaw instrukcji ZX SPECTRUM:

— po pierwsze, należy spowodować wygenerowanie przez system błędu składniowego (ang. syntax error),

— po drugie, „przechwycić" obsługę tego błędu powodując, że zamiast wyprowadzenia komunikatu o błędzie — wykonana zostanie sekwencja instrukcji dająca odmienny, z góry założony efekt.

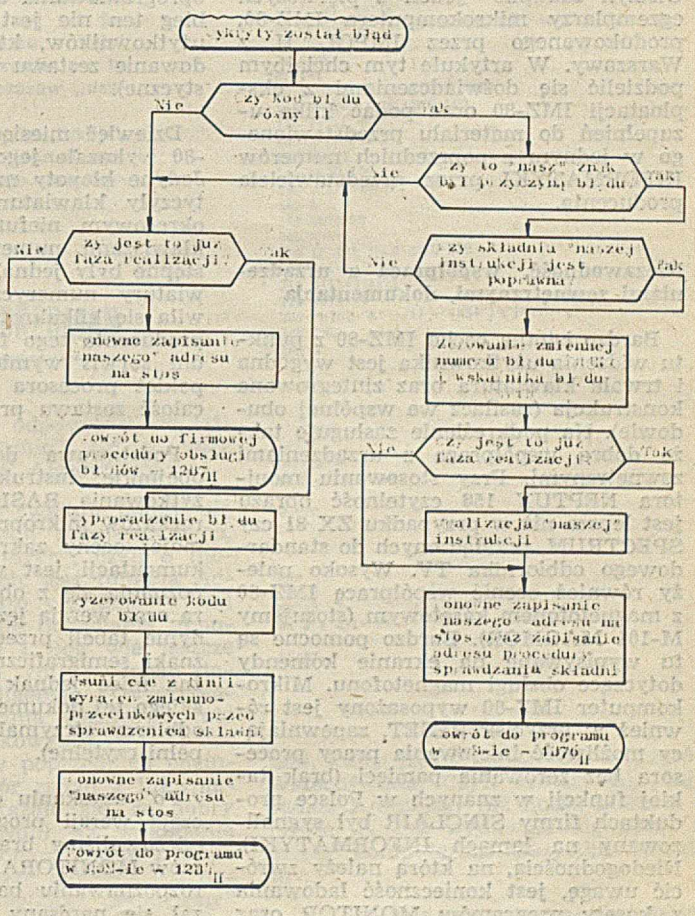
Błąd składniowy można spowodować dwojako (użytkownik ma pełną swobodę wyboru):

A) użycie dowolnego ze słów kluczowych w świadomie błędny sposób (np. DRAW $); nie ma to oczywiście wpływu na normalne użycie i działanie „zniekształcanych" instrukcji w innych miejscach programu.

B) użycie dowolnego ze znaków otrzymywanych przez kombinację z klawiszem SHIFT (np. *, $) i wprowadzenie nazwy nowej instrukcji, litera po literze, np. *OWAL, *ELIPSA. Znaki uzyskiwane przy użyciu klawisza SHIFT służą do wyjścia poza tryb K (tryb wprowadzania instrukcji) po to, aby możliwe było podawanie pojedynczych liter (w trybie K zamiast litery pojawi się od razu całe słowo kluczowe).

Wykrycie nieprawidłowości powoduje przejście do procedury obsługi błędu. Adres tej procedury zapisany jest w lokacji pamięci wskazywanej przez zmienną systemową ERR-SP (adres: 23613/14). Stwierdzenie błędu, zarówno w fazie sprawdzania poprawności składni wprowadzanej instrukcji, jak i w fazie realizacji, powoduje następujące akcje wykonywane przez program zapisany w pamięci ROM:

- przepisanie zawartości zmiennej CH-ADD (adres: 23645) do wskaznika błędu XPTR (adres: 23647); wskaźnik zawiera adres określonej przez interpreter błędnej pozycji w analizowanej linii,
- do zmiennej ERRNR (adres: 23610) zapisywany jest kod błędu wg następującej zasady: numer błędu jest o jeden mniejszy niż numer raportu błędu, który pojawia się po wystąpieniu błędu w fazie realizacji programu; dla numeru błędu 255 występuje raport 0 — O.K., dla błędu numer 11 raport C — nonsense in BASIC; taki właśnie raport pojawiać się będzie przy wykryciu nierozpoznanej przez interpreter instrukcji i to zarówno w fazie sprawdzania poprawności składni jak i w fazie wykonania,
- wskaźnik stosu ładowany jest zawartością zmiennej ERR SP,
- zerowane są zarówno pamięć jak i stos kalkulatora zmiennoprzecinkowego,
- wykonanie instrukcji RET powoduje skok pod adres przechowywany na stosie. Podczas sprawdzania poprawności składni adres ten równy jest 12B7H, zaś znajdujący się tam



Schemat blokowy opisywanego algorytmu

program powoduje wyświetlenie całej linii ze znakiem zapytania na błędnej pozycji. W fazie realizacji programu adres ten równy jest 1303H, zaś znajdujący się tam program spowoduje przerwanie wykonywania programu użytkowego i wyświetlenie odpowiedniego komunikatu błędu.

Umieszczenie na stosie adresu „własnego" programu (powinien być on zapisany powyżej adresu RAMTOP + 1) skieruje działanie systemu operacyjnego do tego właśnie programu, tak jakby wykonywana była standardowa obsługa błędu.

Na rysunku przedstawiony został schemat blokowy algorytmu rozszerzania zestawu instrukcji języka BASIC dla ZX SPECTRUM. W następnej części artykułu zamieszczone zostaną wydruki programów tworzących nowe instrukcje.

MAREK GÓRECKI
Warszawa

Mikroinformatyczny boom zgotowany nam przez firmy polonijne i rzemieślnicze nadal trwa. Zamiast stopniowego zaspokajania głodu możemy jednak obserwować ciągły wzrost apetytów. W rezultacie mikrokomputery są dosłownie rozchwytywane i to na zasadzie: brać to co jest, potem będziemy zastanawiać się co z tym zrobić.

Chociaż zapewne wszyscy zdają sobie sprawę jak zgubne rezultaty może to przynieść, to w obliczu możliwości kupienia za złotówki mikrokomputera tracą głowę. Publikując na łamach mikroKLANU opisy różnych krajowych konstrukcji, liczymy więc bardziej na oddziaływanie podświadomości, która skłoni (lub powstrzyma) do zakupu określonego mikrokomputera do określonych celów. Sądzimy, że oprócz ogólnych opisów dostarczanych nam przez producentów bardzo przydatne będą opinie użytkowników, czyli tych, którzy przymierzyli daną konstrukcję do określonych zadań i przekonali się czy to jest właśnie „ten" komputer jakiego potrzebują.

Zapraszamy do udziału w dyskusji.

Doświadczenia z eksploatacji IMZ-80

W czerwcu ubiegłego roku ZETO Olsztyn zakupiło jeden z pierwszych egzemplarzy mikrokomputera IMZ-80, produkowanego przez IMPOL II z Warszawy. W artykule tym chciałbym podzielić się doświadczeniami z eksploatacji IMZ-80 oraz podać kilka uzupełnień do materiału przedstawionego w jednym z poprzednich numerów INFORMATYKI przez przedstawiciela producenta.

Niezawodność, współpraca z urządzeniami zewnętrznymi, dokumentacja

Bardzo istotną cechą IMZ-80 z punktu widzenia użytkownika jest wygodna i trwała klawiatura oraz zintegrowana konstrukcja (zasilacz we wspólnej obudowie). Na podkreślenie zasługuje także dobra współpraca z urządzeniami zewnętrznymi. Przy stosowaniu monitora NEPTUN 156 czytelność obrazu jest lepsza niż w przypadku ZX 81 czy SPECTRUM, przyłączonych do standardowego odbiornika TV. Wysoko należy również ocenić współpracę IMZ-80 z magnetofonem kasetowym (stosujemy M-101 MAGMOR). Bardzo pomocne są tu wypisywane na ekranie komendy dotyczące obsługi magnetofonu. Mikrokomputer IMZ-80 wyposażony jest również w przycisk RESET, zapewniający możliwość inicjowania pracy procesora bez zerowania pamięci (brak takiej funkcji w znanych w Polsce produktach firmy SINCLAIR był sygnalizowany na łamach INFORMATYKI). Niedogodnością, na którą należy zwrócić uwagę, jest konieczność ładowania z kasety programów MONITOR oraz BASIC. Jednak producent zapowiedział już możliwość wpisania podstawowego oprogramowania do pamięci stałej (zabieg ten nie jest potrzebny dla tych użytkowników, którzy planują rozbudowanie zestawu IMZ-80 o dyski elastyczne).

Dziewięć miesięcy eksploatacji IMZ-80 wykazało jego dużą niezawodność. Jedyne kłopoty natury technicznej dotyczyły klawiatury. Polegały one na okresowym niefunkcjonowaniu linii z klawiszami numerycznymi (cyfry dostępne były jednak nadal na polu klawiatury numerycznej). Usterka pojawiła się kilkakrotnie. Po zgłoszeniu do producenta tego faktu w ciągu dwóch dni serwis wymienił klawiaturę oraz pakiet procesora i od tego momentu całość zestawu pracuje bez zarzutu.

Podstawowa dokumentacja IMZ-80 obejmuje: instrukcję obsługi, opis użytkowania BASIC IMZ-80 oraz listę rozkazów mikroprocesora Z-80. Według mojej oceny zakres merytoryczny dokumentacji jest wystarczający do zapoznania się z obsługą mikrokomputera oraz wersją języka BASIC (brak jedynie tabeli przedstawiającej dostępne znaki semigraficzne). Zastrzeżenia można mieć jednak do poziomu edytorskiego tej dokumentacji (może następni odbiorcy otrzymali egzemplarze już w pełni czytelne).

Po uzyskaniu od producenta „roboczej" wersji programu ASSEMBLER odczuwaliśmy brak opisów podprogramów MONITORA. Przy ich dokładnym rozpoznawaniu bardzo przydatny okazał się napisany przez nas wcześniej (w języku BASIC) DISASSEMBLER. Pomocą służył również konstruktor IMZ-80. Pierwsze miesiące eksploatacji były dla firmy IMPOL II i ZETO Olsztyn okresem zbierania i wymiany doświadczeń. M.in. na podstawie naszych sugestii producent wprowadził kilka usprawnień do oprogramowania IMZ-80. Ostatnio wraz z „handlową" wersją programu ASSEMBLER IMZ-80 otrzymaliśmy pełną dokumentację obejmującą opisy programów ASSEDIT, COMPILER i DEBUGGER oraz istotnych podprogramów zawartych w programie MONITOR. Załącznikiem jest również lista rozkazów mikroprocesora Z-80 w porządku alfabetycznym i szesnastkowym (w takim formacie działa DEBUGGER).

Mylące jest użycie przez przedstawiciela producenta w swoim artykule nazwy MACROASSEMBLER, w odniesieniu do zestawu trzech programów zapewniających możliwość tworzenia i testowania oprogramowania w języku ASSEMBLER. Wprawdzie możliwe jest stosowanie pewnej grupy pseudorozkazów (m.in. zapewniających wygodną organizację pamięci), ale w sumie nazwa MACROASSEMBLER jest użyta na wyrost. Tworzenie programów na tym poziomie byłoby szybsze, gdyby przynajmniej funkcje ASSEDIT i COMPILER zawarte były w jednym programie.

BASIC IMZ-80

Dla czytelników mikroKLANU będzie chyba dogodne porównanie istotnych elementów języka BASIC IMZ-80 z implementacją tego języka dla ZX SPECTRUM.

W IMZ-80 wygodniejszy jest proces tworzenia, a przede wszystkim poprawiania programów. Do tego celu służy edytor ekranowy umożliwiający dokonywanie zmian w programie poprzez operacje na tekście wyświetlonym na ekranie. Po przekroczeniu 40 znaków w wierszu, automatycznie dołączony zostaje następny wiersz zapewniając możliwość tworzenia programów w logicznych liniach zawierających maksymalne 79 znaków. W procesie tworzenia i testowania programów można korzystać z kilku dyrektyw, które nie występują w ZX SPECTRUM: AUTO (automatyczne numerowanie wierszy z zadanym krokiem), DELETE n-m (kasowanie fragmentu programu), TRON (śledzenie przebiegu programu — wypisywanie na ekran numerów kolejno wykonywanych linii).

Poniżej publikujemy informację o wydarzeniu, które może okazać się przełomowe dla rozwoju mikroinformatyki lub też... zaginąć w pomroce dziejów. Czas pokaże. Na razie możemy zauważyć (z pozycji kibica — mikroKLAN nie jest przecież klubem), że publikowana deklaracja ma wiele cech wspólnych z tym co pisaliśmy wprowadzając wkładkę mikroKLAN. Nie to jest jednak źródłem aprobaty jaką darzymy nową inicjatywę. Jesteśmy przekonani, że jedynym sposobem na uchronienie mikroinformatyki od możliwych dewiacji (jakim uległo wiele innych dziedzin naszego życia) jest ruch masowy. Kluby i zrzeszająca je Federacja wydają się na dziś jedyną możliwą formą takiego ruchu.

Polska Federacja Klubów Mikroinformatycznych

„Mikroinformatyka" to pojęcie nie zdefiniowane jeszcze w żadnym słowniku języka polskiego. A jednak przenika ono coraz trwalej w sferę naszej świadomości jako nowe, odrębne od dotychczasowych zjawisko. Fenomen mikroinformatyki bez wątpienia ma swoje źródła w mikroelektronice i tradycyjnej informatyce, wymaga jednak również uwzględnienia, że jest to zarazem masowo upowszechniana metoda konstruktywnego rozumowania.

Chociaż mikroinformatyka w Polsce to ciągle jeszcze w dużej mierze nowa moda, lecz z rodzaju tych, na których może bardzo wiele zyskać całe społeczeństwo.

Zaleganie na tyłach światowego postępu niesie ze sobą również pewną szansę, jak dotąd nie dość docenianą, na zmarnowanie której w obecnej sytuacji po prostu nas nie stać. Śledząc drogę światowego rozwoju mikrokomputeryzacji powinniśmy zwrócić uwagę na wszelkie popełnione dotąd błędy tak aby wybrać najlepszą z tysiąca możliwych ścieżek. Konstruktywnym przejawem troski o wybór pomyślnej drogi rozwoju mikroinformatyki w naszym kraju jest zamysł utworzenia POLSKIEJ FEDERACJI KLUBÓW MIKROINFORMATYCZNYCH.

Federacja jako porozumienie i dobrowolne zrzeszenie wszelkich powstających na terytorium Polski klubów mikrokomputerowych byłaby w stanie przełamać barierę informacyjną (poprzez dysponowanie własnym bankiem informacji) co ma kluczowe znaczenie dla zapewnienia prawidłowego rozwoju mikroinformatyki.

W założeniach, Federacja jako reprezentantka interesów klubów (czyli grup społecznych) i inicjatorka ich powstawania realizowałaby postawione sobie cele poprzez organizowanie spotkań, pokazów, seminariów i kursów oraz wszelkie inne dostępne środki przekazywania informacji.

29 czerwca 1985 roku wybrano tymczasowe władze Federacji (nie jest ona jeszcze oficjalnie zarejestrowana). Wybory generalne odbędą się około 29 listopada 1985 w czasie trwania „Dni Klubów Mikroinformatycznych" w ramach MIKROEXPO '85.

dokończenie na str. 18

Podstawowy kurs Akademii mikroKLANU zmierza ku końcowi. Czy dołożyliśmy własną cegiełkę do rozwijania umiejętności programowania — czas pokaże. Został nam jeszcze jeden odcinek a potem... zobaczymy. Wszystko zależy od reakcji Czytelników. Były już wprawdzie postulaty kursu... języka C, ale jest to chyba propozycja przedwczesna — zbyt mało mikrokomputerów w naszym kraju „rozumie" ten język. Wśród innych propozycji wymieniany był też język LOGO — bez wątpienia bardzo ciekawy. Sądzimy jednak, że tu z kolei sensowna byłaby znacznie szersza popularyzacja (szkoły!), niż pozwala na to skromny nakład INFORMATYKI. Odłożymy więc chyba LOGO do czasów „usamodzielnionego" mikroKLANU — kiedyś to w końcu musi nastąpić.

Nie zamierzamy oczywiście zerwać z publikacjami dotyczącymi języka BASIC. Nadal oczekujemy na ciekawe procedury i podprogramy. Mamy jednak nadzieję, że będą one pisane w eleganckim stylu, tak jak to zalecano „na wykładach" Akademii.

Akademia mikroKLANU (7) ACKERMAN

```
LIST

5 REM OBLICZANIE FUNKCJI ACKERMANA METODA LISTY CELOW.
10 DIM J(1000)
20 INPUT "J=";J(1)
30 IF J(1)<0 OR INT(J(1))<>J(1) THEN GOTO 20
40 INPUT "K=";K
50 IF K<0 OR INT(K)<>K THEN GOTO 40
60 J1=J(1)
70 KS=K
80 I=1
90 IF J(I)=0 THEN GOTO 180
100 IF K=0 THEN GOTO 150
110 I=I+1
120 J(I)=J(I-1)
130 K=K-1
140 GOTO 100
150 J(I)=J(I)-1
160 K=1
170 GOTO 90
180 K=K+1
190 IF I=1 THEN GOTO 230
200 I=I-1
210 J(I)=J(I)-1
220 GOTO 90
230 PRINT "A(";J1;",";KS;")=";K
240 END
READY.
```

```
RUN
J=? 2
K=? 23
A( 2 , 23 )= 49

READY.
```

```
RUN
J=? 3
K=? 2
A( 3 , 2 )= 29

READY.
```

Wprawdzie w języku BASIC nie mamy możliwości rekurencyjnego definiowania funkcji, ale możemy tę trudność ominąć używając tzw. listy celów. Tworzymy sekwencję par liczb j_i, k gdzie każda para jest kolejnym celem obliczeń. Gdy $i = 1$ oraz $j_i = 0$, wtedy policzyliśmy już wszystkie potrzebne cele i znamy odowiedź końcową czyli k. Oczywiście konkretna postać listy celów wynika z algorytmu Ackermana. Najlepiej ilustruje to program szkolny (wydruk).

Jeżeli puścimy nasz program, to już dla niewielkich liczb ($j = 4$, $k = 2$) otrzymamy błąd przekroczenia wartości wskaźnika wektora J. Zwiększamy więc wymiar wektora J, aż do granic pamięci naszego komputera (np. DIM J(100000)). Rezultat — ten sam tyle, że po bardzo, bardzo długim czasie! A przecież dla niektórych liczb otrzymywaliśmy poprawne odpowiedzi, co łatwo sprawdzić „ręcznie". Gdzie więc pies pogrzebany? W Algorytmie!

Nasza lista celów świetnie nadaje się do obliczania funkcji rekurencyjnych, ale takich „normalnych", czyli o niewielkich wartościach. Zaś funkcja Ackermana jest bardzo perfidna — jest to jakby uogólnienie dodawania, mnożenia, potęgowania i Bóg wie jeszcze czego! Łatwo się o tym przekonać stosując własną inteligencję i wyprowadzając tzw. równania różnicowe (nie mylić z różniczkowymi) — patrz wspomniana już w poprzednich odcinkach książka „Zbiór zadań z informatyki", str. 227. Okazało się, że dobry algorytm nie pozwolił na wykonanie obliczeń w skończonym czasie i w ograniczonym obszarze pamięci. Tu nasuwa się uwaga ogólna: „skończony" w dziedzinie programowania, to znaczy taki, który jest mniejszy od wymiarów Wszechświata. Czyli krótszy niż wiek świata i mniejszy niż liczba atomów w tymże. A już wydawało się nam, że dzięki komputerom wszystko potrafimy...

JAKUB TATARKIEWICZ

W mikroKLANIE 9 zamieściliśmy test pozwalający w pewnym zakresie wnioskować o dokładności numerycznej komputera. Dodajmy, że w stosunkowo ograniczonym zakresie, gdyż jak wykazało życie (a raczej czytelnicy w telefonach do redakcji) — wyłożył się na nim choćby minikomputer z serii PDP, dając zaskakująco słabe wyniki. Czy to oznacza, że PDP-y zostały już pokonane przez mikrokomputery? Chyba jeszcze nie całkiem. Każdy test jest w pewnym zakresie subiektywny — coś uwypukla, coś zaniedbuje. Nie należy jednak załamywać rąk: jeśli wykonujemy obliczenia numeryczne, powinniśmy w jakiś sposób szacować rzetelność wyników. Pozostaje więc... doskonalić testy. Tym razem prezentujemy test standardowych funkcji. Czytelników, którzy wymyślili lub znają lepsze testy, zapraszamy do podzielenia się z innymi za pośrednictwem naszych łamów.

Test numeryczny

Publikujemy test, którego głównym celem jest porównanie dokładności obliczeń numerycznych standardowych funkcji. Pomysł zaczerpnęliśmy z listu Martena den Boera do redakcji PHYSICS TODAY (listopad 1984, str. 154), nieco modyfikując zarówno sam test, jak i obliczanie wartości **D**.

A oto lista programu:

```
10 P2 = 1.57
12 DEF FN A(X) = ATN (X / ( SQR ( - X * X + 1) + 1E - 35))
13 DEF FN B(X) = 2 * ATN (1) - FN A(X)
20 INPUT "ILOSC ITERACJI";N
30 D = 0.0
40 A = 0.0
50 S = P2 / N
60 FOR I = 1 TO N
70 A = A + S
80 B = SIN (A)
90 C = FN A(B)
100 B = COS (C)
110 C = FN B(B)
120 B = TAN (C)
130 C = ATN (B)
140 B = EXP (C)
150 C = LOG (B)
160 B = SQR (C)
170 C = B ^ 2
180 D = D + (A - C) * (A - C)
190 NEXT I
200 D = SQR (D / N)
210 PRINT "DOKLADNOSC = ";D
220 END
```

Test zapisany dla APPLE II Europlus w języku MBASIC

Przepis na uruchomienie: Wpisz program do pamięci Twego komputera. Jeżeli BASIC, którego używasz, nie ma funkcji arcsin oraz arccos, to użyj linii 12 oraz 13, a linie 90 i 110 pozostaw bez zmiany. Jeżeli natomiast dostępna Ci wersja języka BASIC ma wspomniane funkcje trygonometryczne, to wstaw je w liniach 90 i 110, a opuść linie 12 oraz 13.

Wpraw w ruch program dla **N = 17**, otrzymany czas (w sekundach) odpowiada przybliżonemu czasowi (w minutach), jaki otrzymasz dla **N = 1000**. Przeprowadź tekst dla **N = 1000** i zanotuj wynik (czas oraz wartość **D**). Porównaj swój komputer z tabelą.

Wnioski potrafi wyciągnąć chyba każdy Czytelnik mikroKLANU: MERITUM nadaje się do gier, a IBM PC z koprocesorem — do obliczeń naukowych. Chyba że posiadamy MERITUM i musimy odwrócić macierz 45×45, czego Czytelnikom mikroKLANU nie życzy

JAKUB TATARKIEWICZ

Autor serdecznie dziękuje wszystkim, którzy udostępnili swoje komputery dla przeprowadzenia powyższego testu.

Wyniki testu dla różnych mikrokomputerów

Komputer	Język	Czas (s)	D	Uwagi
IBM PC	MBASIC	1 7	4.139E-06	sic!
	TURBO PASCAL	10,7	8.632E-16	koprocesor 8087
	turbo pascal	439	2.836E-10	
TI 99/4A	Extended BASIC	1318	8.013E-12	
HP 85	HP BASIC	362	1.149E-11	ma ASN i ACS
HP 75C	HP BASIC	423	1.149E-11	ma ASN i ACS
HP 41C	RPN	3252	1.869E-09	ma ASN i ACS
SINCLAIR ZX81	BASIC	957	8.828E-09	ma ASN i ACS
	BASIC	4020	8.828E-09	fast mode slow mode
SINCLAIR SPECTRUM	BASIC	995	8.818E-09	ma ASN i ACS
	PASCAL	69	2.948E-05	
APPLE II+	MBASIC	732	1.145E-08	
	MBASIC kompilowany	502	1.147E-08	
	PASCAL	837	1.561E-06	
NEPTUNE 184	BASIC	835	1.145E-08	
CBM 64	BASIC	835	1.145E-08	
VIC 20	BASIC	678	1.148E-08	
EPSON HX-20	MBASIC v.1.0	965	1.606E-06	
MERITUM	BASIC	623	1.609E-06	brak DEF FN
TRS 80/I	BASIC (level II)	1020	1.609E-06	
	FORTRAN	400	4.733E-06	nieoptymalizowany
OSBORNE	BASIC	333	4.628E-06	
Twój komputer	.....	.....	.....	.....

dokończenie ze str. 17

Pierwszą zakrojoną na szerszą skalę imprezą organizowaną przez Federację mają być DNI KULTURY MIKROINFORMATYCZNEJ MIKROEXPO '85. Ten swego rodzaju chrzest bojowy ma być sprawdzianem sensu istnienia Federacji a przede wszystkim ludzi i gotowych do współpracy organizacji, czyli wszystkich nieobojętnych na stan mikroinformatyki w Polsce.

Według wstępnych planów przewidziane zostały następujące prezentacje: grafika komputerowa, muzyka komputerowa, krajowa i zagraniczna oferta sprzętowa, dorobek poszczególnych klubów, filmy o aktualnym stanie mikroinformatyki na świecie (EXPO 85).

Planowane jest też zorganizowanie cyklu seminaryjnego, w którym m.in. odbędzie się spotkanie na temat języka LOGO, jako narzędzia do celów edukacyjnych i profesjonalnych. Będą też tematy związane z prezentacjami.

Do udziału w MIKROEXPO '85 zaproszeni będą goście z krajów RWPG, Europy Zachodniej i Stanów Zjednoczonych. W czasie Dni Kultury opracowany zostanie raport o stanie polskiej kultury mikroinformatycznej. Podjęta zostanie w nim próba ukazania roli różnych instytucji w dotychczasowym rozwoju mikroinformatyki.

Pragniemy zaapelować do wszystkich, którzy działają w różnych klubach mikrokomputerowych (lub mających zamiar w ten sposób się zorganizować) o aktywne przyłączenie się do naszych działań. Mamy nadzieję, że dzięki powstaniu Federacji wiele trudności przestanie być problemem, a dostęp do różnych typów mikrokomputerów nie będzie nieosiągalną, niejasną w czasie i przestrzeni mrzonką, lecz realną możliwością dla każdego zainteresowanego.

PIOTR TYMOCHOWICZ
ul. Renesansowa 5 m. 10
01-905 Warszawa, tel. 35-77-60